Freescale Semiconductor
Application Note

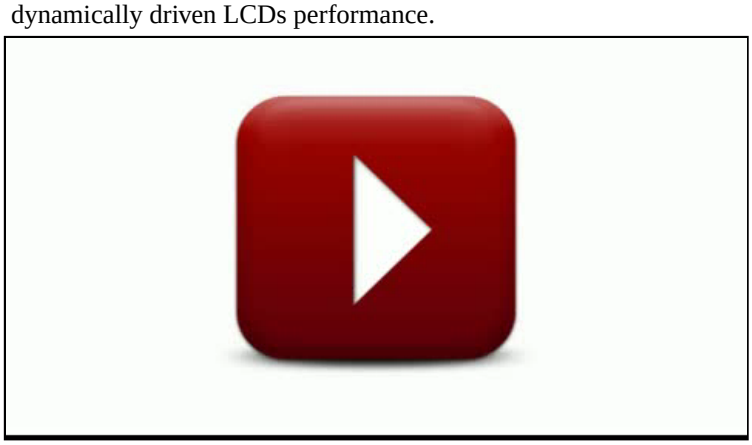
Document Number: AN3412
Rev. 0, 4/2010

Dynamic LCD Driver Using GPIO Pins

by: Juan Cazares
Applications Engineer
Mexico
Guadalajara

1 Introduction

This application note explains how the twisted nematic (TN) liquid crystal display (LCD) works and how this kind of LCD can be controlled using the microcontroller's GPIO pins. There are two ways to control the TN LCD, statically and dynamically. The driver discussed in this document is intended to drive dynamic LCDs. The dynamically driven LCDs require a pure AC voltage waveform applied to the LCD electrodes. Each electrode needs one backplane electrode and one frontplane terminal to be displayed. A dynamic LCD makes the situation more complicated because the segments are organized into a matrix, but then again this kind of LCD reduces the number of terminals used. The Freescale application note titled *XGATE Library: TN/STN LCD Driver Driving Bare TN and STN LCDs Using GPIO Pins* (document AN3219) explains in depth the dynamically driven LCDs performance.



Contents

2 LCD Overview

To reduce the number of terminals used to drive the LCD, the segments are multiplexed. This means each terminal drives two or more segments. The terminals are also called frontplanes. The amount of segments driven depends on the number of LCD backplanes. The LCD manufacturer must specify the number of backplanes used to drive the LCD. This driver is intended to manage a dynamically driven LCD with 4 backplanes, although the number of backplanes can be changed. The number of backplanes is also called a duty ratio, in this case ¼ duty ratio.

The LCD model used in this application note is the VIM-878-DP. Each digit in the display is made up of 16 segments in total. The 16 segments of each digit can be driven with four terminals, this is because the segments are organized into a matrix. Figure 1 and Figure 2 show how the display segments are organized.

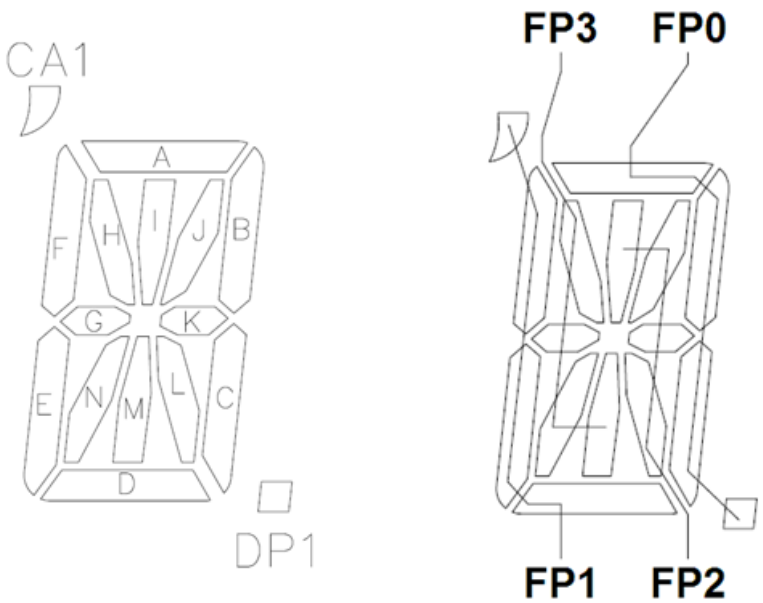


Figure 1. Display segments

To polarize one segment a differential voltage must exist between the desired frontplane terminal and the appropriate backplane. Table 1 shows the backplane and frontplane connections required to polarize the 16 segments shown in Figure 1.

Table 1. Backplane and frontplane connections

	Backplane0	Backplane1	Backplane2	Backplane3
Frontplane0	DP	C	B	A
Frontplane1	D	E	F	CA
Frontplane2	L	K	J	I
Frontplane3	M	N	G	H

The polarity of the voltage waveform applied to each segment must be inverted periodically to maintain a constant electric field between electrodes and to avoid damage on segments because of the DC component. All the visible segments must be refreshed with a minimal refreshing frequency of 50 Hz or 60 Hz. The software driver provided in this application note (AN3412SW) refreshes each segment with a frequency equal to 400 Hz. The entire LCD is refreshed at 100 Hz. This is faster than the minimal frequency needed to enhance the contrast control implemented by the software.

To turn segment C on; frontplane0 and backplane1 must be enabled. The same frontplane drives four segments, the rest of the frontplanes and backplanes must remain disabled. To turn segments C and E on; frontplane0, frontplane1, and backplane1 must be enabled.

To turn segments C and D on; frontplane0, frontplane1, backplane0, and backplane1 must be enabled. If both frontplanes and backplanes are enabled at the same time the segments DP, C, D, and E are displayed. This means that two or more backplanes can not be enabled at the same time.

To turn on all the segments they must be enabled in four phases. In phase one, the segments under the backplane0 can be displayed, in phase two, the segments under backplane1 can be displayed and so on. When the four phases are refreshed the cycle must be repeated.

Dynamically driven LCDs require more than two voltage levels to drive segments. The different voltage levels to drive a dynamic LCD must be at least 0, 1.6, and 3 volts, so the required voltage levels are generated with external resistors. The voltage levels are called bias, in this case ½ bias. To turn any segment on, the voltage applied between the desired frontplane terminal and the appropriate backplane must be 3 volts. To turn any segment off, the voltage level applied between the frontplane and backplane terminal must not be higher than 1.6 volts.

3 Hardware

This driver was tested with the M68EVB08GB60 Rev. C, the XTWR-MCF51MM Rev. A, and the XTWR-MCF51JE Rev. A evaluation boards. The LCD driver uses GPIO pins to drive the backplanes and frontplanes making it easier to migrate the driver to any microcontroller or any custom board. The hardware connections between the MCU and the LCD are described in Table 2 and Table 3.

Table 2. M68EVB08GB60 Rev.C and LCD connections

MCU Pin	LCD Pin Number	COM0	COM1	COM2	COM3
PTA5	1	1D	1E	1F	CA1
PTA6	2	1L	1K	1J	1I
PTA1	3	2D	2E	2F	CA2
PTA2	4	2L	2K	2J	2I
PTC5	5	3D	3E	3F	CA3
PTC6	6	3L	3K	3J	3I
PTC1	7	4D	4E	F4	CA4
PTC2	8	4L	4K	4J	4I
PTD5	9	5D	5E	5F	CA5
PTD6	10	5L	5K	5J	5I
PTD1	11	6D	6E	6F	CA6
PTD2	12	6L	6K	6J	6I
PTF5	13	7D	7E	7F	CA7
PTF6	14	7L	7K	7J	7I
PTF1	15	8D	8E	8F	CA8
PTF2	16	8L	8K	8J	8I
PTG4	17	COM0	—	—	—
PTG5	18	—	COM1	—	—
PTG6	19	—	—	COM2	—
PTG7	20	—	—	—	COM3
PTF0	21	DP8	8C	8B	8A
PTF3	22	8M	8N	8G	8H

MCU Pin	LCD Pin Number	COM0	COM1	COM2	COM3
PTF4	23	DP7	7C	7B	7A
PTF7	24	7M	7N	7G	7H
PTD0	25	DP6	6C	6B	6A
PTD3	26	6M	6N	6G	6H
PTD4	27	DP5	5C	5B	5A
PTD7	28	5M	5N	5G	5H
PTC0	29	DP4	4C	4B	4A
PTC3	30	4M	4N	4G	4H
PTC4	31	DP3	3C	3B	3A
PTC7	32	3M	3N	3G	3H
PTA0	33	DP2	2C	2B	2A
PTA3	34	2M	2N	2G	2H
PTA4	35	DP1	1C	1B	1A
PTA7	36	1M	1N	1G	1H

Table 3. XTWR-MCF51MM Rev.A, XTWR-MCF51JE Rev.A, and LCD connections

MCU Pin	LCD Pin Number	COM0	COM1	COM2	COM3
PTJ5	1	1D	1E	1F	CA1
PTJ6	2	1L	1K	1J	1I
PTF5	3	2D	2E	2F	CA2
PTF6	4	2L	2K	2J	2I
PTD5	5	3D	3E	3F	CA3
PTD6	6	3L	3K	3J	3I
PTC1	7	4D	4E	4F	CA4
PTC2	8	4L	4K	4J	4I
PTH1	9	5D	5E	5F	CA5
PTH2	10	5L	5K	5J	5I
PTH5	11	6D	6E	6F	CA6
PTH6	12	6L	6K	6J	6I
PTJ1	13	7D	7E	7F	CA7
PTJ2	14	7L	7K	7J	7I
PTE1	15	8D	8E	8F	CA8
PTE2	16	8L	8K	8J	8I
PTA1	17	COM0	—	—	—
PTG5	18	—	COM1	—	—
PTB6	19	—	—	COM2	—
PTG7	20	—	—	—	COM3

MCU Pin	LCD Pin Number	COM0	COM1	COM2	COM3
PTE0	21	DP8	8C	8B	8A
PTE3	22	8M	8N	8G	8H
PTJ0	23	DP7	7C	7B	7A
PTJ3	24	7M	7N	7G	7H
PTH4	25	DP7	6C	6B	6A
PTH7	26	6M	6N	6G	6H
PTH0	27	DP5	5C	5B	5A
PTH3	28	5M	5N	5G	5H
PTC0	29	DP4	4C	4B	4A
PTC3	30	4M	4N	4G	4H
PTD4	31	DP3	3C	3B	3A
PTD7	32	3M	3N	3G	3H
PTF4	33	DP2	2C	2B	2A
PTF7	34	2M	2N	2G	2H
PTJ4	35	DP1	1C	1B	1A
PTJ7	36	1M	1N	1G	1H

This driver is intended to work with ½ bias LCDs, therefore only the backplanes need pull-up and pull-down resistors to generate waveforms. The resisitor value is important because it controls the DC component. The LCD manufacturer specifies the maximum DC component that can be applied to the LCD. For this application note the value for all pull-up and pull-down resistors is equal to 15 K ohms. These values generate a maximum of 16 mV on the DC component. See Figure 2.

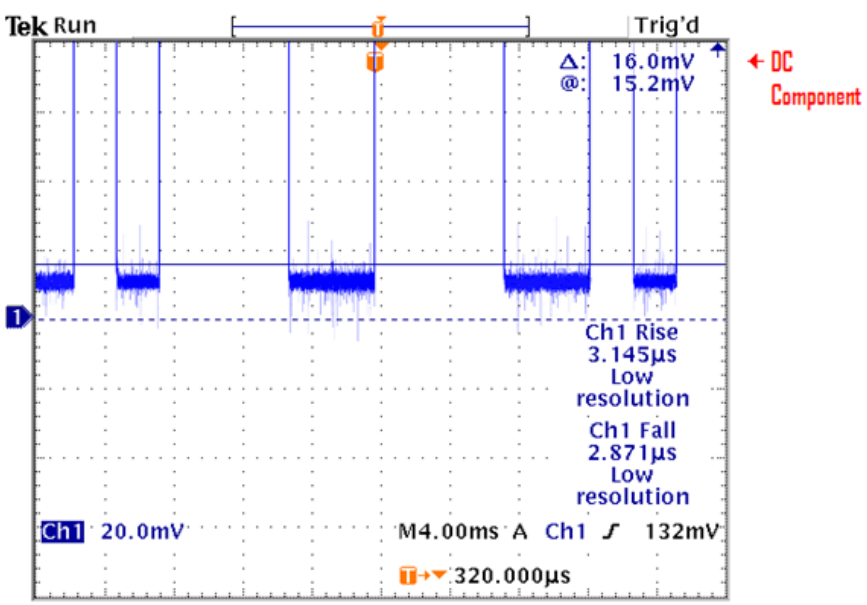


Figure 2. Maximum DC component applied to the LCD electrodes

The block diagram shows how the MCU and the LCD must be connected. Notice that only the backplane nets have pull-up and pull-down resistors.

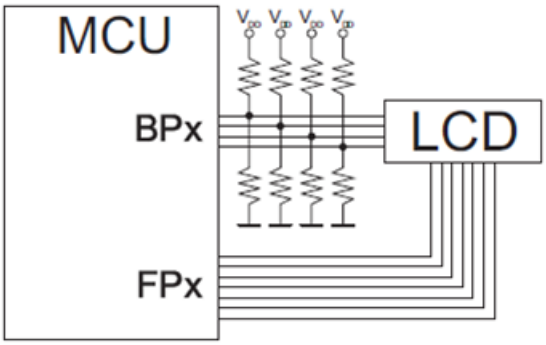


Figure 3. Block diagram

The following schematic shows a way to connect the TN LCD to the XTWR-MCF51MM Rev.A, and the XTWR-MCF51JE Rev.A. The schematic and layout files are provided with this application note.

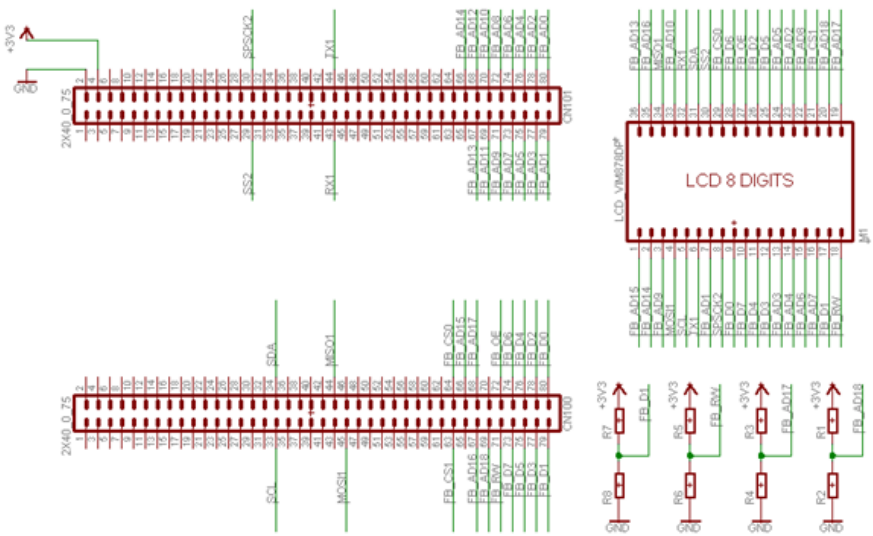


Figure 4. TN LCD schematic

4 Functional Description

The AN3412SW source code for the LCD driver can be downloaded from www.freescale.com. The driver is made up of two files, the LCD.c and LCD.h. Including these files into an application the user can perform any of the capabilities described in the following sections.

4.1 Configuring the Driver

The driver configuration is performed by modifying the macro definitions located in the LCD.h header file.

4.2 Driver Initialization

After the MCU comes out of reset, all GPIO pins used to drive the backplanes and frontplanes must be configured. The time base to refresh the LCD segments must also be enabled, to do this, the function `vfnLCDInit` must be called by the main application.

4.3 Driver Control

The driver function `vfnLCDDriver` refreshes the backplanes and frontplanes when the refresh cycle period elapses. This function must be called into the main loop to get the driver working properly.

4.4 LCD Refresh Cycle Frequency

Each frontplane and backplane must be refreshed periodically. This period of time is generated with a timer comparator, therefore a timer channel must be configured as output compare. In this case Timer2 Channel0 is used to generate an interrupt each 2.5 milliseconds. The waveform period can be calculated using the following formula.

$$\mathrm{WaveformPeriod} = (\mathrm{DesiredPeriod} * \mathrm{BusFrequency}) / \mathrm{TimerPrescaler}$$

The timer and channel used to refresh the LCD can be changed if desired. To change the LCD the next macro definition in the LCD.h file must be modified.

```
#define WAVE_FORM_PERIOD                    (390)
     #define CONSTRAST_HIGH                          (WAVE_FORM_PERIOD/2)
     #define CONSTRAST_MED                           (WAVE_FORM_PERIOD)
     #define CONSTRAST_LOW                           (WAVE_FORM_PERIOD*2)

     #define LCD_INTERRUPT_VECTOR            (9)
     #define TIMER_REGISTER                            (TPM2SC)
     #define TIM_CHANNEL_REGISTER             (TPM2C0SC)
     #define RELOAD_TIMER_VALUE                (TPM2C0V)
     #define CLEAR_TIMER_FLAG                      (TPM2C0SC_CH0F = 0)
```

4.5 Contrast Control

The driver has an LCD contrast adjustment that does not need an external component. The contrast can be adjusted by modifying the value stored in the variable `g16ContrastValue`. When increasing the value stored in `g16ContrastValue`, the contrast decreases. And if decreasing the value stored then the contrast increases. The contrast value can be changed at any moment and the value can be from 0–65535.

4.6 Maximum Number of Digits

Although the driver is intended to drive an eight digit LCD, the number of digits handled can be changed by modifying the macro definition `MAX_DIGITS` located in the LCD.h header file. Each of the digits are managed with four frontplanes, that is four GPIO pins. Any desired port can be used to drive any digit in the LCD. The unique restriction is each frontplane must be part of the same port and must be aligned by nibbles.

The following macros configure what GPIO pins drive digit1 in the LCD. This macro is using the most significant nibble of port A to drive digit1.

```
     #define LCD_DIGIT1_NIBBLE                      (HIGH_NIBBLE)
     #define LCD_DIGIT1_PORT_INIT                 (PTADD |= LCD_DIGIT1_NIBBLE)
```

```
     #define LCD_DIGIT1_CLEAR_PORT          (PTAD &= ~LCD_DIGIT1_NIBBLE)
     #define LCD_DIGIT1_PORT                         (PTAD)
```

For example, if digit1 is managed with the least significant nibble from port E, the macro definitions must be modified as follows.

```
#define LCD_DIGIT1_NIBBLE                  (LOW_NIBBLE)
     #define LCD_DIGIT1_PORT_INIT               (PTEDD |= LCD_DIGIT1_NIBBLE)
     #define LCD_DIGIT1_CLEAR_PORT              (PTED &= ~LCD_DIGIT1_NIBBLE)
     #define LCD_DIGIT1_PORT                    (PTED)
```

The source code provided with this application note provides the necessary macros to drive eight digits. To add more digits you must copy and paste the macros mentioned above and replace the desired port and nibble that will be used to drive the new digit.

4.7 Configuring Backplanes

Each one of the four backplanes can be configured individually. The backplanes can be managed with different MCU ports if desired. The following macro definition configures which GPIO pin is used to drive backplane0.

```
     #define BACK_PLANE0_DDR                    (PTGDD_PTGDD4)
     #define BACK_PLANE0_PIN                      (PTGD_PTGD4)

     #define BACK_PLANE0_DDR                    (PTGDD_PTGDD4)
     #define BACK_PLANE0_PIN                      (PTGD_PTGD4)
```

4.8 LCD Buffer

The polarity of the voltage waveform applied to each segment must be inverted periodically. To do so the driver works with the values stored in the buffer named `LCDBuffer`. The length of the buffer must be equal to the maximum number of digits. In this case the `LCDBuffer` length is eight.

To display a character in the digit1 the appropriate value must be stored in the `LCDBuffer` element 0. To display a character in the digit2 the appropriate value must be stored in `LCDBuffer` element 1 and so on.

4.9 Displaying Characters

As shown in Figure 1 each digit is formed with 16 segments. These are not characters generated by default, so any number, letter, or symbol must be designed by the user. The driver is provided with an array that allows displaying numbers and uppercase letters. The array is called `gau16CharactersArray` and it is located in the LCD.c source file. To display numbers on the LCD, the desired number value can be passed directly as the array index. The following example uses the array of characters to display the numbers 01800999 on the LCD.

```
LCDBuffer[0].u16Word = gau16CharactersArray[0];
     LCDBuffer[1].u16Word = gau16CharactersArray[1];
     LCDBuffer[2].u16Word = gau16CharactersArray[8];
     LCDBuffer[3].u16Word = gau16CharactersArray[0];
     LCDBuffer[4].u16Word = gau16CharactersArray[0];
     LCDBuffer[5].u16Word = gau16CharactersArray[9];
     LCDBuffer[6].u16Word = gau16CharactersArray[9];
     LCDBuffer[7].u16Word = gau16CharactersArray[9];

     for(;;)
     {
          __RESET_WATCHDOG();                          /* feeds the dog */

          vfnLCDDriver();                              /* Call LCD Driver */
     }
```

To display a letter, the desired letter value in ASCII format must be subtracted with a value equal to 30 hexadecimal (zero in ASCII format) before passing it as the array index. The following example uses the array of characters to display the string HELLO which begins in digit4.

```
LCDBuffer[0].u16Word = BLANK_DIGIT;
    LCDBuffer[1].u16Word = BLANK_DIGIT;
    LCDBuffer[2].u16Word = BLANK_DIGIT;
    LCDBuffer[3].u16Word = gau16CharactersArray['H' – '0'];
    LCDBuffer[4].u16Word = gau16CharactersArray['E' – '0'];
    LCDBuffer[5].u16Word = gau16CharactersArray['L' – '0'];
    LCDBuffer[6].u16Word = gau16CharactersArray['L' – '0'];
    LCDBuffer[7].u16Word = gau16CharactersArray['O' – '0'];

    for(;;)
    {
         __RESET_WATCHDOG();                          /* feeds the dog */

         vfnLCDDriver();                              /* Call LCD Driver */
    }
```

4.10 Creating New Characters

The user can generate characters or symbols by modifying the `gau16CharactersArray` array. Each element in the array must have only the segments that need to be visible. If a character or symbol needs several segments to be displayed, each one of the visible segments must be added with an OR operation. For example to display the letter " B " the segments that must be visible are: A, B, C, D, E, F, and K.

The corresponding element in the array must be as follows:

```
(unsigned int)(SEG_A|SEG_B|SEG_C|SEG_D|SEG_E|SEG_F|SEG_K), //B
```

Any desired symbol can be added to the array of characters but you must be responsible for providing the correct definitions.

4.11 Displaying Message

In some cases the eight available digits on the LCD are not enough to display large messages. To solve this issue, the driver provides a function that allows displaying any message regardless the length of the message. The function is named `vfnLCDPrintMessage` and receives two parameters. The first parameter (`u8pTextSource`) is a pointer that points to the beginning of the message you want displayed. The second parameter (`u8TopDigit`) is the length of the message. If the value stored in `u8TopDigit` is higher than eight, the message is shifted from right to left. When the entire message is displayed the action is repeated again. If the value stored in `u8TopDigit` is lower or the same as eight, the message remains static.

Each time the function is called by the main application the message is shifted one digit. This means that the shifting frequency depends on how fast the function `vfnLCDPrintMessage` is called.

The following example displays the first 20 elements in the array named TestMsg. The shifting frequency is given by the value of the variable `u32Counter` and the bus frequency.

```
 unsigned long u32Counter = 20000;

for(;;)
{
          __RESET_WATCHDOG();                          /* feeds the dog */

          vfnLCDDriver();                              /* Call LCD Driver */

          if(!(--u32Counter))
          {
               u32Counter = 20000;
               vfnLCDPrintMessage(&TestMsg[0],20);
          }
 }
```

The function described above can be used to display a static message if the function is called only once. The next example displays the first six elements of the message stored in TestMsg.

```
 vfnLCDPrintMessage(&TestMsg[0],6);

for(;;)
{
     __RESET_WATCHDOG();                    /* feeds the dog */

     vfnLCDDriver();                        /* Call LCD Driver */
}
```

4.12 Code Size and Execution Time

Table 4. Required memory size (bus frequency = 20 MHz)

Driver Functions	Code Size (bytes)	Data Size (bytes)	Execution Time (µS)
VfnLCDInit	50	0	5.2
VfnLCDDriver	198	1	9.2
vfnLCDPrintMessage	134	5	100
LCD_ISR	64	1	3.8

Table 5. File sizes

File	Data (bytes)	Code (bytes)	Constant (bytes)
LCD.c	39	762	116
C language overhead	126	391	19
Stack (default)	80	—	—
Entire project	245	1153	135

Table 6. Performance details (bus frequency = 20 MHz)

Performance Detail	Value	Unit
Backplanes slew rate delay	4.4	uS
Maximum DC component	16	mV
Backplanes resistors	15	K ohms
Maximum current consumption x segment	12	nA

5 Conclusion

This application note shows that anyTN LCD can be managed by using general purpose pins. The unique restriction is the number of GPIO pins available on the microcontroller.

How to Reach Us:

Home Page:
www.freescale.com

Web Support:
http://www.freescale.com/support

USA/Europe or Locations Not Listed:
Freescale Semiconductor
Technical Information Center, EL516
2100 East Elliot Road
Tempe, Arizona 85284
+1-800-521-6274 or +1-480-768-2130
www.freescale.com/support

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
www.freescale.com/support

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064
Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor China Ltd.
Exchange Building 23F
No. 118 Jianguo Road
Chaoyang District
Beijing 100022
China
+86 10 5879 8000
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor Literature Distribution Center
1-800-441-2447 or +1-303-675-2140
Fax: +1-303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

Document Number: AN3412
Rev. 0, 4/2010